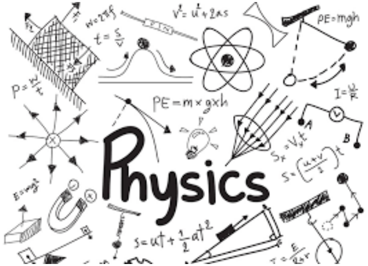
What Google knows about ML languages that you may not

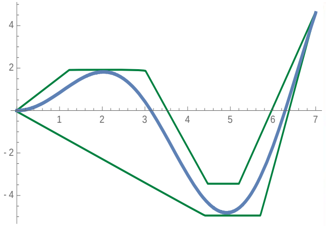
(Spoiler: only Swift and Julia make the cut)

Alan Edelman (MIT & JC)
Viral Shah (Julia Computing)
Juan Pablo Vielma (MIT)
Chris Rackauckas (UC Irvine)

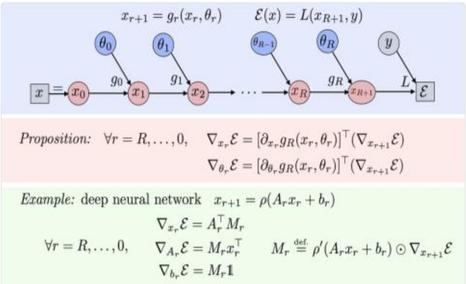
Software Toolchain



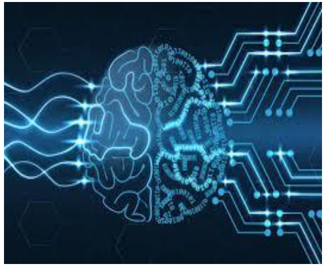
Multiphysics (PDEs)



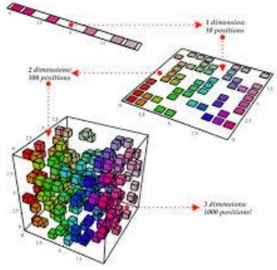
Optimization



Adjoint Methods
(Backprop/Autodiff)



Machine
Learning

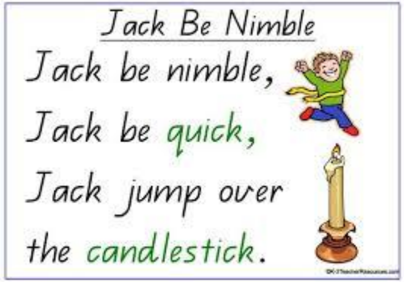


Surrogate Models
Dim Reduction

Composability

Sensitivity Analysis
Confidence Intervals

Performance
Nimble/Agile



Uncertainty
Quantification

?

Scalable

Modern Software Development

Languages Studied by Google as Powerful for ML

https://github.com/tensorflow/swift/blob/master/docs/WhySwiftForTensorFlow.md

R

python™

JavaScript

TypeScript

RUST

Ruby PROGRAMMING Language

Scala

Go

Java™

julia

Swift Programming Language

Languages Studied by Google as Powerful for ML

1. Filter on Technical Merits

Languages Studied by Google as Powerful for ML

julia

Swift Programming Language

2. Filter on Usability

1. Filter on Technical Merits

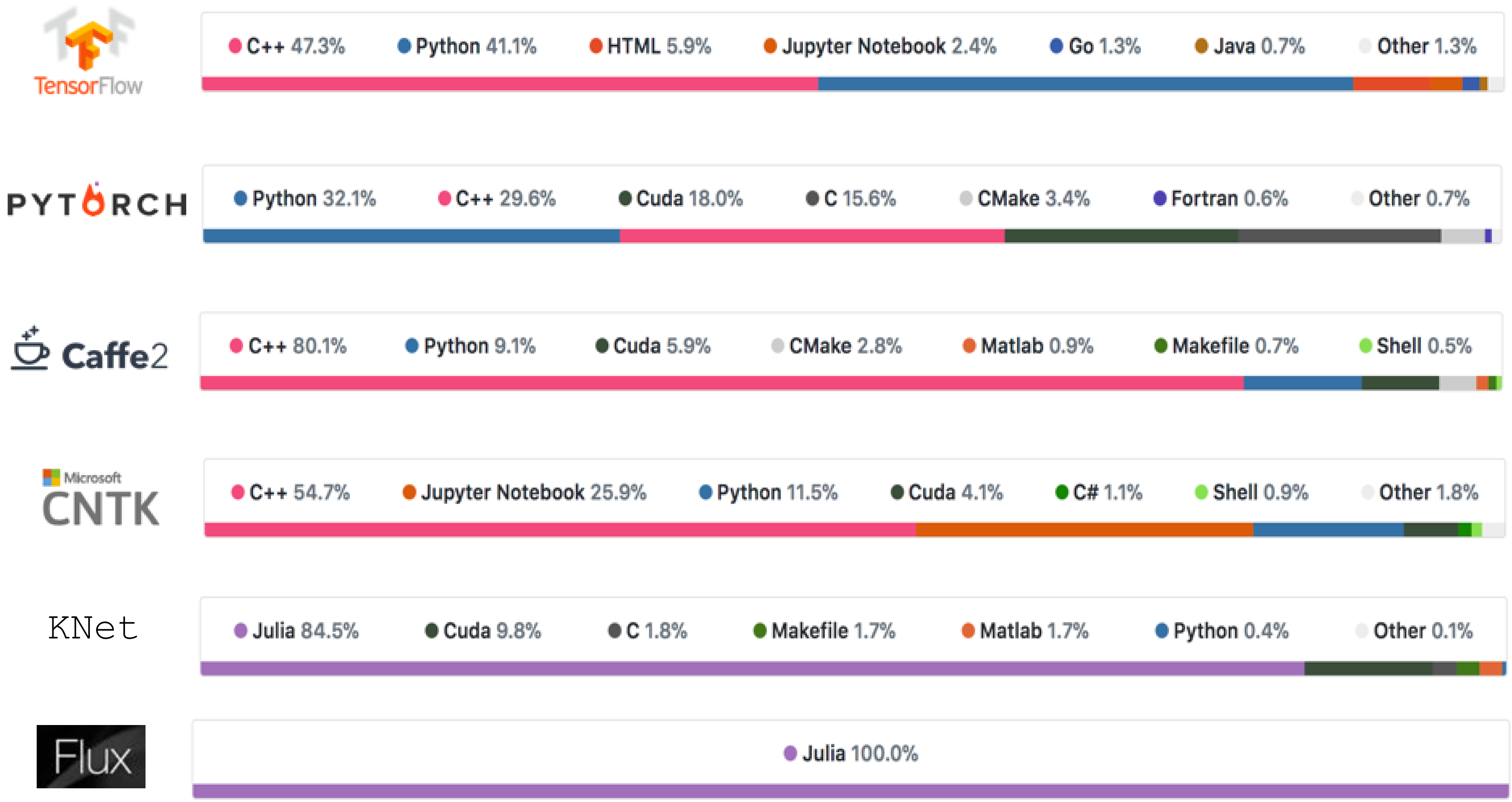
Languages Studied by Google as Powerful for ML

Julia: Julia is … currently investing in machine learning techniques, and even have good interoperability with Python APIs. **The Julia community shares many common values…**

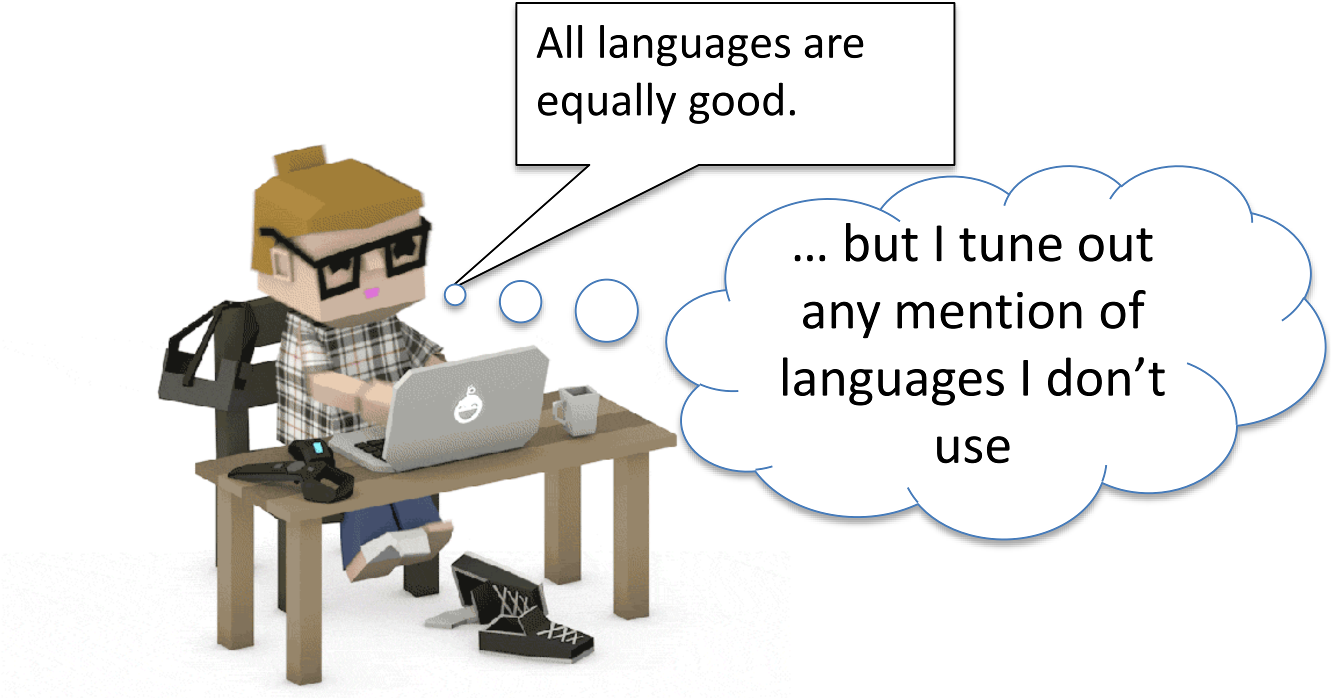
Swift Programming Language

2. Filter on Usability

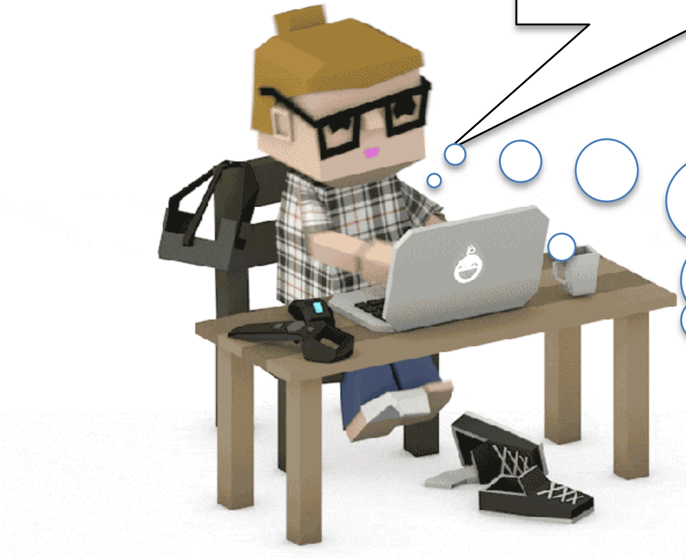
1. Filter on Technical Merits


TensorFlow
C++ 47.3%
Python 41.1%
HTML 5.9%
Jupyter Notebook 2.4%
Go 1.3%
Java 0.7%
Other 1.3%
PYTORCH
Python 32.1%
C++ 29.6%
Cuda 18.0%
C 15.6%
CMake 3.4%
Fortran 0.6%
Other 0.7%
Caffe2
C++ 80.1%
Python 9.1%
Cuda 5.9%
CMake 2.8%
Matlab 0.9%
Makefile 0.7%
Shell 0.5%
Microsoft
CNTK
C++ 54.7%
Jupyter Notebook 25.9%
Python 11.5%
Cuda 4.1%
C# 1.1%
Shell 0.9%
Other 1.8%
KNet
Julia 84.5%
Cuda 9.8%
C 1.8%
Makefile 1.7%
Matlab 1.7%
Python 0.4%
Other 0.1%
Flux
Julia 100.0%

Psychology of Programming Languages



Psychology of Programming Languages

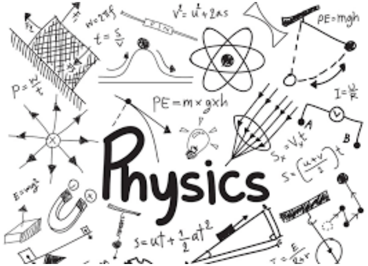


All languages are equally good.

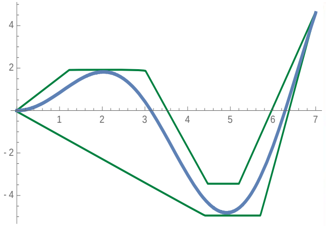
… but I tune out any mention of languages I don't use

…at least until Google tells me to pay attention

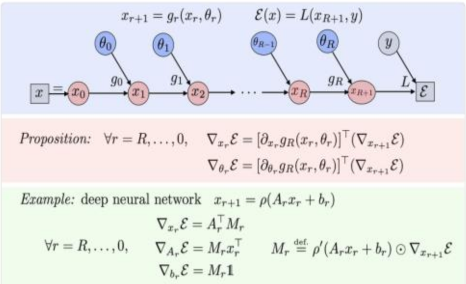
Software Toolchain



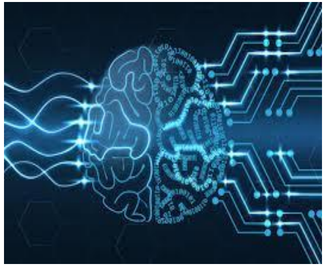
Multiphysics (PDEs)



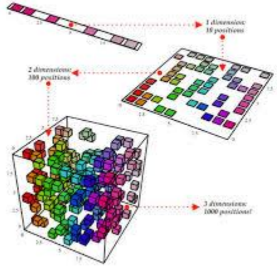
Optimization



Adjoint Methods
(Backprop/Autodiff)



Machine
Learning

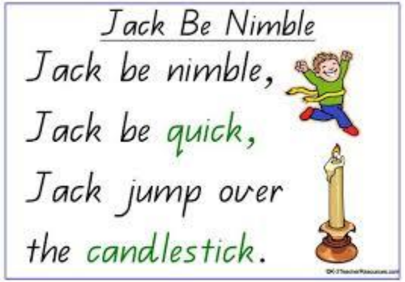


Surrogate Models
Dim Reduction

Composability

Sensitivity Analysis
Confidence Intervals

Performance
Nimble/Agile

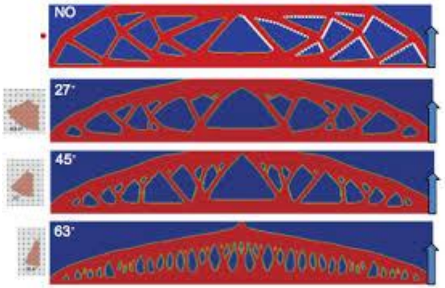


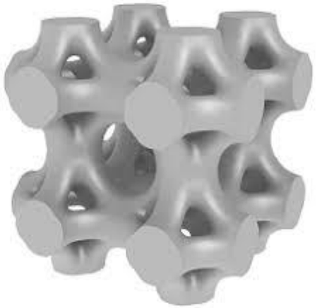
Uncertainty
Quantification

?

Scalable

Topology Optimization ---- 20 years ago what we did





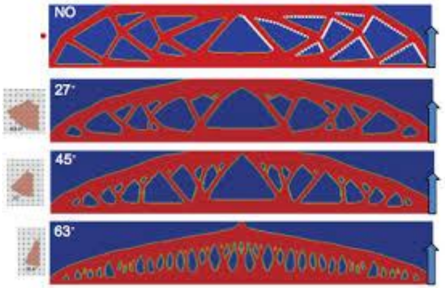
Write Force Balance Law → Finite Elements → Linear System → Solve → Graph

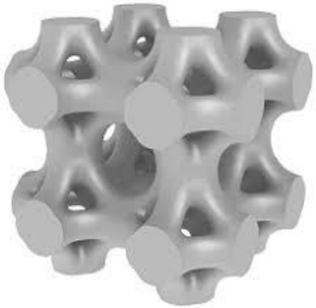
Dense Matrices

↓

Sparse Matrices

Now!





Fancy Differential Equations → Dimensionality Reduction →

Write Force Balance Law → Finite Elements → Linear System → Solve → Graph

Topology Optimizaton

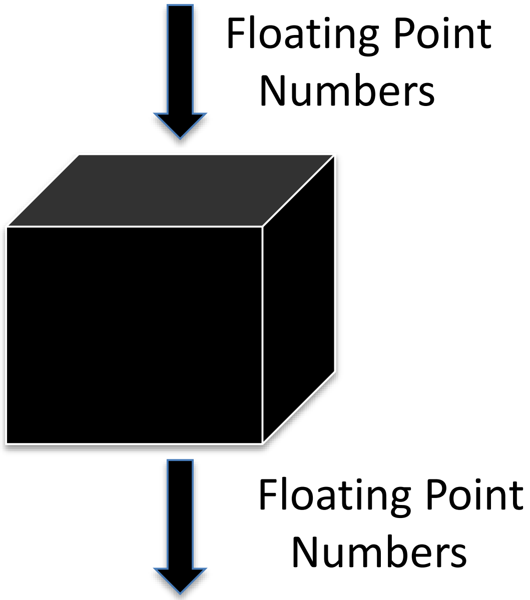
Dense Matrices

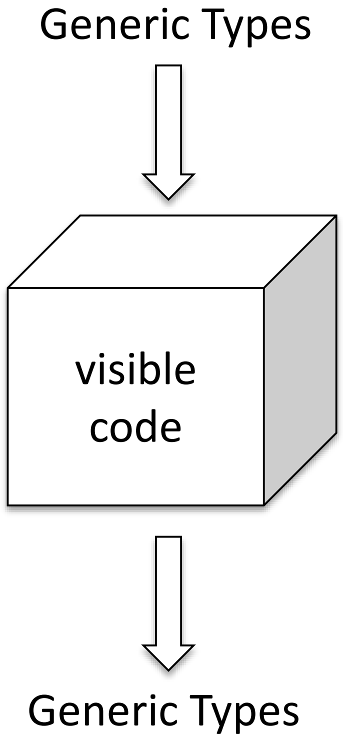
↓

Sparse Matrices

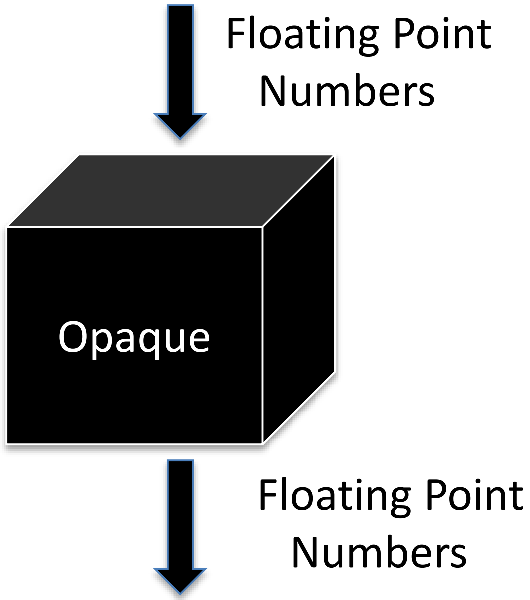
Compose many physical systems

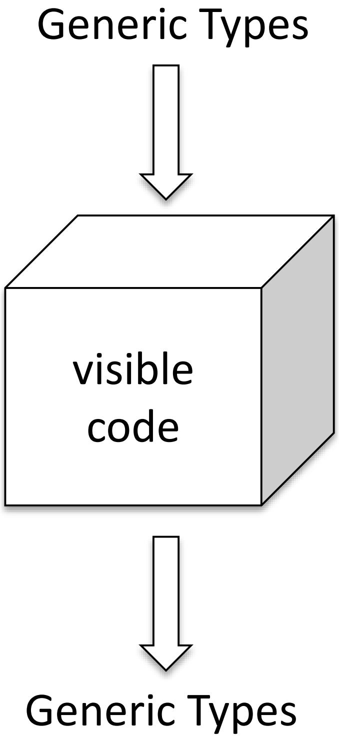
Black Boxes vs White Boxes





Black Boxes vs White Boxes





☺ Legacy Code

☹ Can’t hit all the criteria

Code must rewrite other code

An Idealized Modern Toolchain for Energy

(we can have this!)

Today (Fragmented)

What could be with high level tools & generic types

PDEs, PDMPs, …

True Physical Equations

floats in

floats out

Neural Network, …

Surrogate Model

floats in

floats out

Optimization

floats in

floats out

← trigger on demand, sensitivity analysis

adaptive data generation for more accurate surrogates

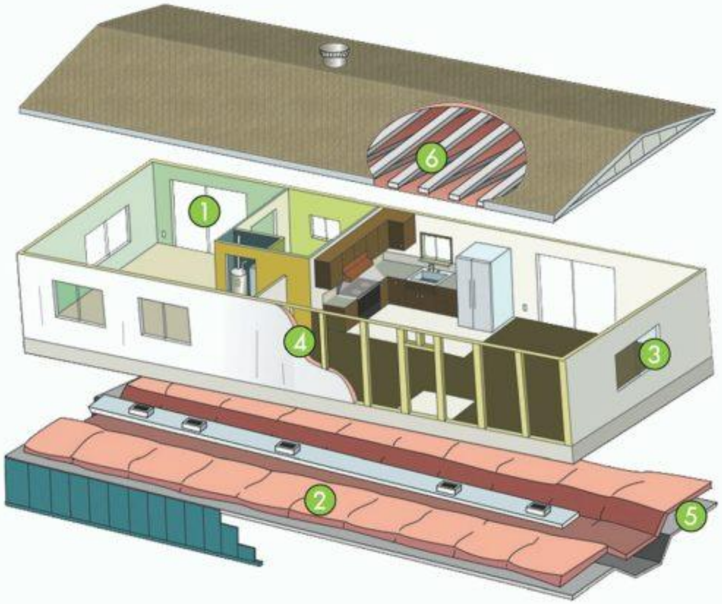
← needs programmable form

specialized machine learning models for efficient optimization

optimal solutions with uncertainty estimates

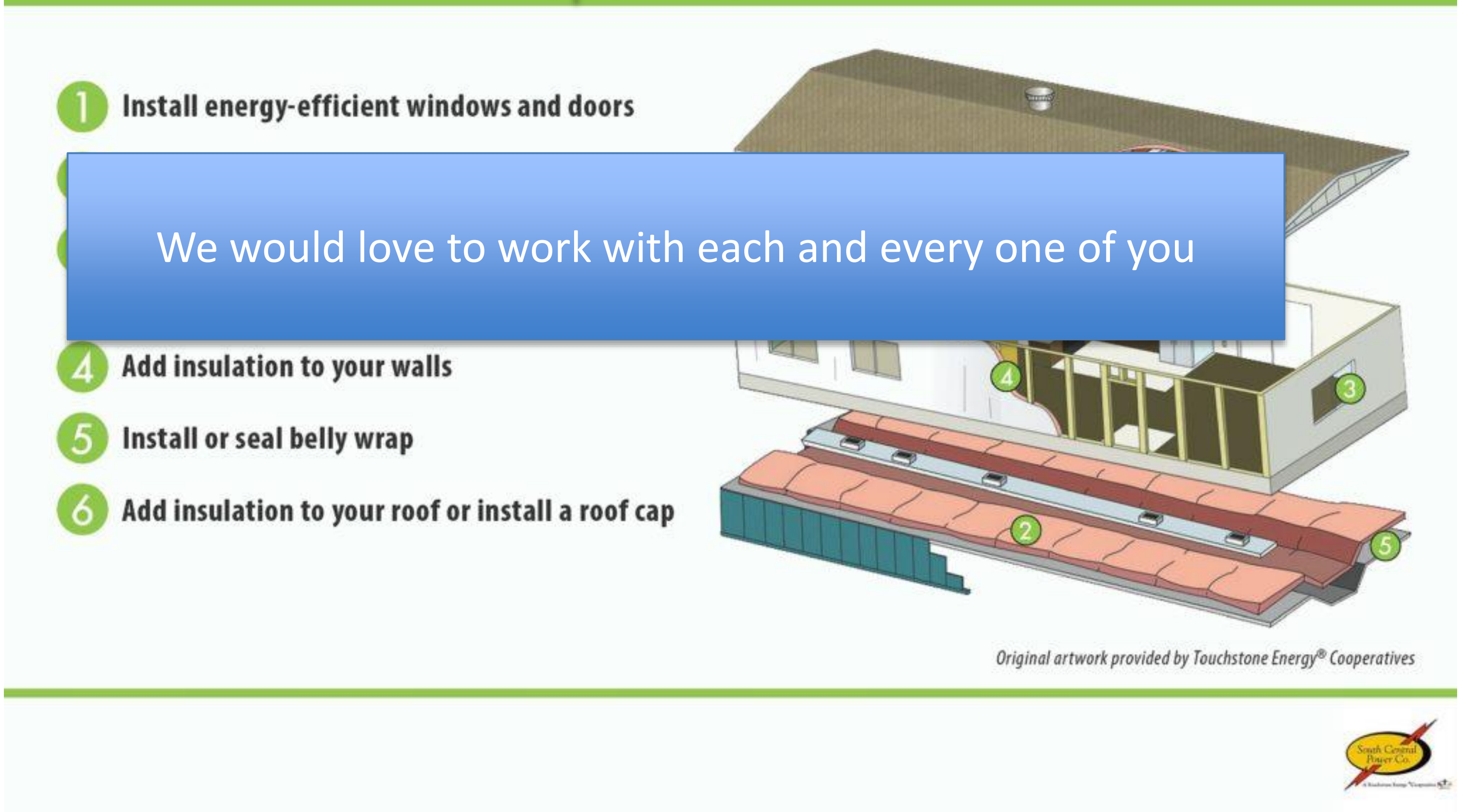
RETROFITTING YOUR MANUFACTURED HOME FOR ENERGY EFFICIENCY

1. Install energy-efficient windows and doors
2. Replace insulation in the belly
3. Make general repairs (seal bottom board, caulk windows, doors, ducts, etc.)
4. Add insulation to your walls
5. Install or seal belly wrap
6. Add insulation to your roof or install a roof cap



Original artwork provided by Touchstone Energy® Cooperatives

Retrofitting your software for machine learning, sensitivity analysis, scalability, optimization



We would love to work with each and every one of you

ML models are really programs

- Support hardware accelerators (GPUs, TPUs, Nervana, New silicon)
- Parallelization (Multi-threading, Multi-GPU, Distributed)
- Optimization (Placement, Memory Use, Low overhead)
- Automatic Differentiation
- Ease of programming (Math notation, Debuggers, Libraries)
- Ease of deployment (Cloud, Phones, Embedded)

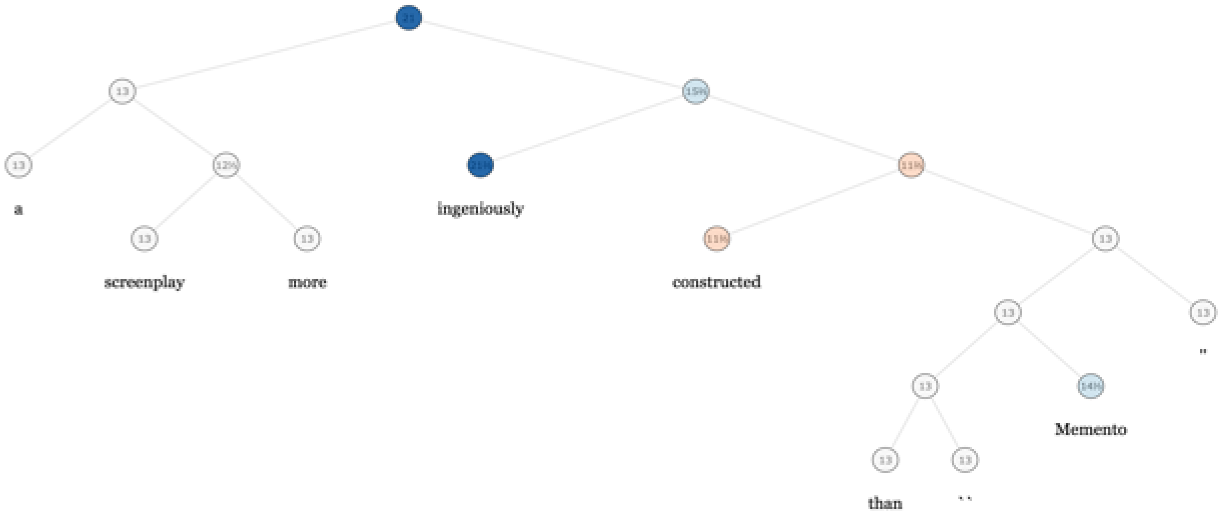
ML problems are really language problems

New models have new demands

Models commonly need:

- Conditional branching
- Loops for recurrence
- Recursion over trees

```
# Stanford TreeBank
def model(tree):
 if isleaf(tree):
   tree.value
 else:
   model(tree.left) + model(tree.right)
```



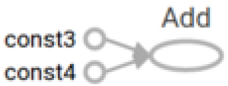
In areas such as probabilistic programming

- Models need to reason about *other* programs (e.g. program generators and interpreters)
- Include non-differentiable components like Monte Carlo Tree Search.

TensorFlow is more like a language and less like a library

- We build a “computational graph” (essentially an AST)

```
a = tf.constant(3.0, dtype=tf.float32)
b = tf.constant(4.0) # also tf.float32 implicitly
total = a + b
print(a)
print(b)
print(total)
```



Lazy (Eval) programming in JS

```
function add(a,b) {
    return `${a}+${b}`;
}

x = 1; y = 2
z = add('x', 'y') // 'x+y'
eval(z) // 3
x = 4
eval(z) // 6
```

- Which may contain control flow (tf.if, tf.while), variable scoping

```
def my_image_filter(input_images):
    with tf.variable_scope("conv1"):
        # Variables created here will be named "conv1/weights", "conv1/biases".
        relu1 = conv_relu(input_images, [5, 5, 32, 32], [32])
    with tf.variable_scope("conv2"):
        # Variables created here will be named "conv2/weights", "conv2/biases".
        return conv_relu(relu1, [5, 5, 32, 32], [32])
```

- Cannot reuse existing libraries. Need new libraries for I/O and data processing.

Abadi, M., Isard, M., Murray, D., *A Computational Model for TensorFlow: An Introduction*, 2018.

Mixed Integer Optimization and Julia

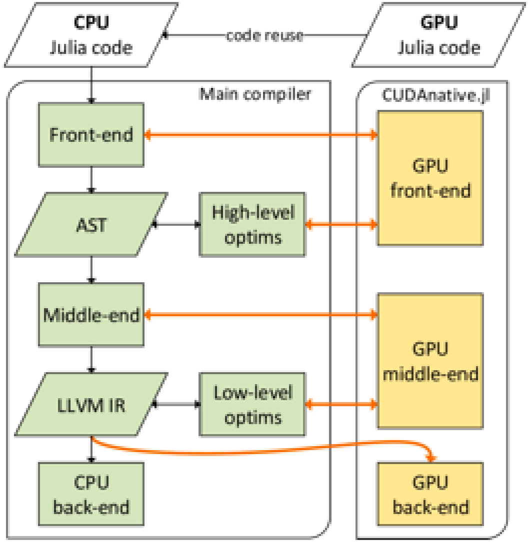
- Mixed Integer Optimization
  - Discrete + nonlinear
  - Theoretically hard
  - Routinely solved in practice

http://www.gurobi.com/company/example-customers

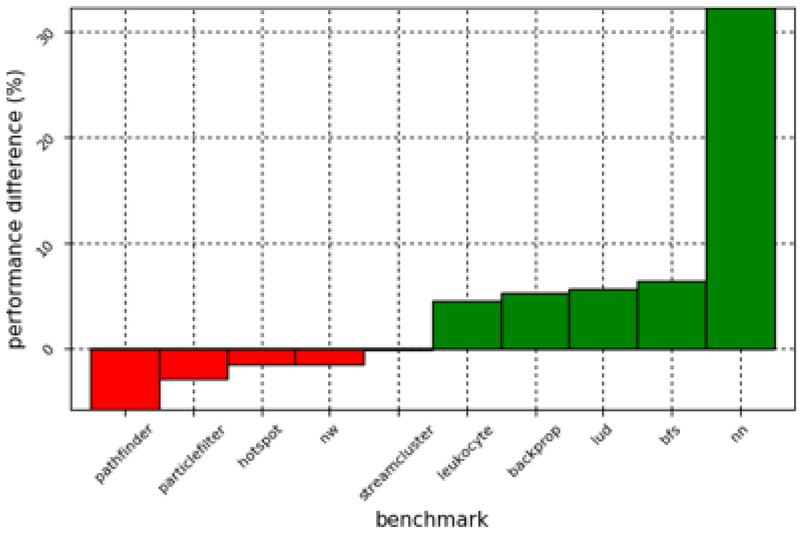
- JuMP
  - Optimization modelling language and interphase
  - Easy to use and advanced
  - Integrated into Julia

GPU computing in Julia

Native Array Libraries – CuArrays.jl, GPUArrays.jl, CLArrays.jl



CUDAnative.jl: 1,300 LOC



Performance difference between CUDA C++ and CUDAnative.jl implementations of several benchmarks from the Rodinia benchmark suite.

Besard. T., Foket, C., De Sutter, B., *Effective Extensible Programming: Unleashing Julia on GPUs*, 2017.

Julia ML at PetaScale to catalog the visible universe

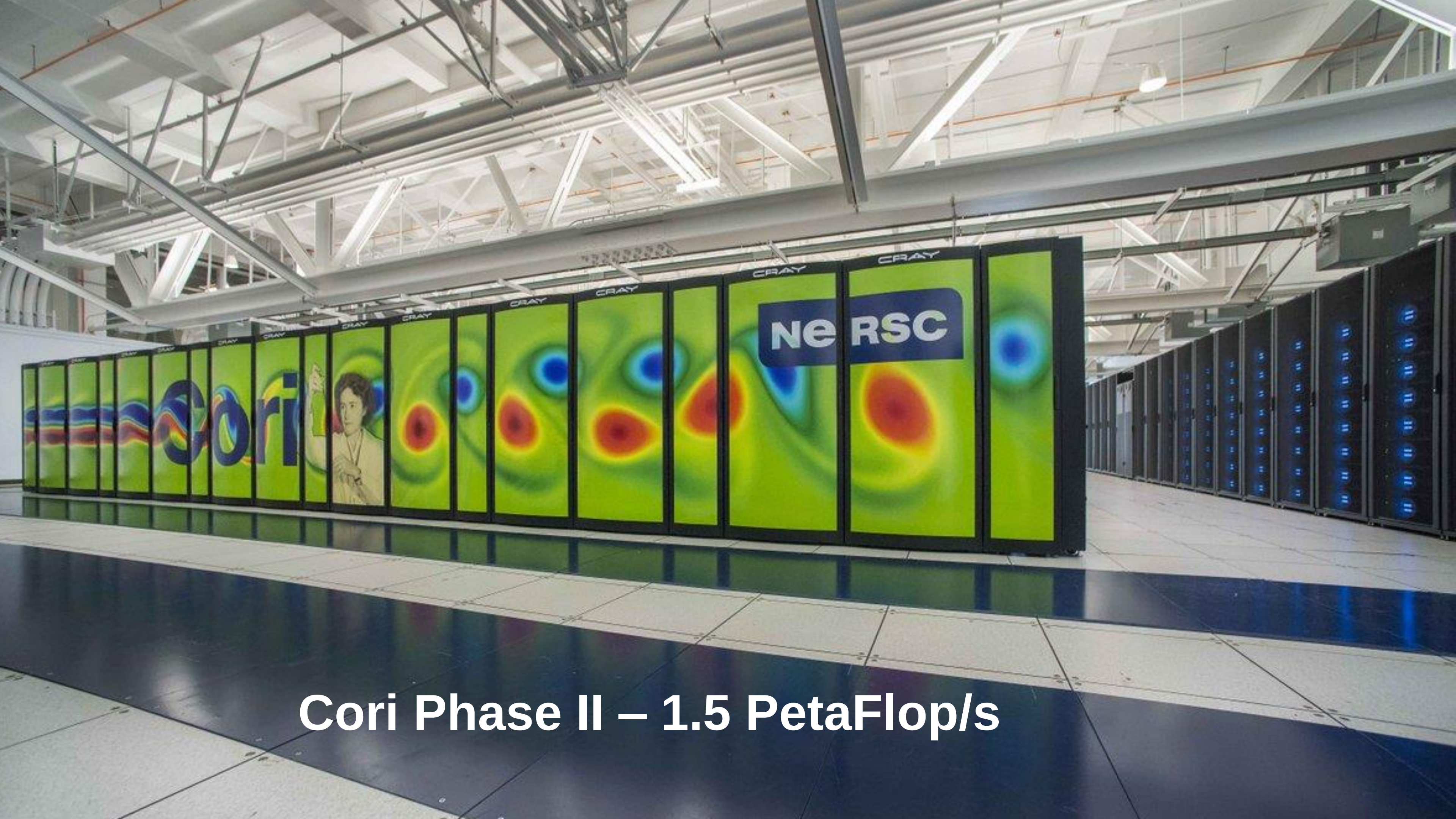
650,000 cores. 1.3M threads. 60 TB of data.

Most light sources are near the detection limit.

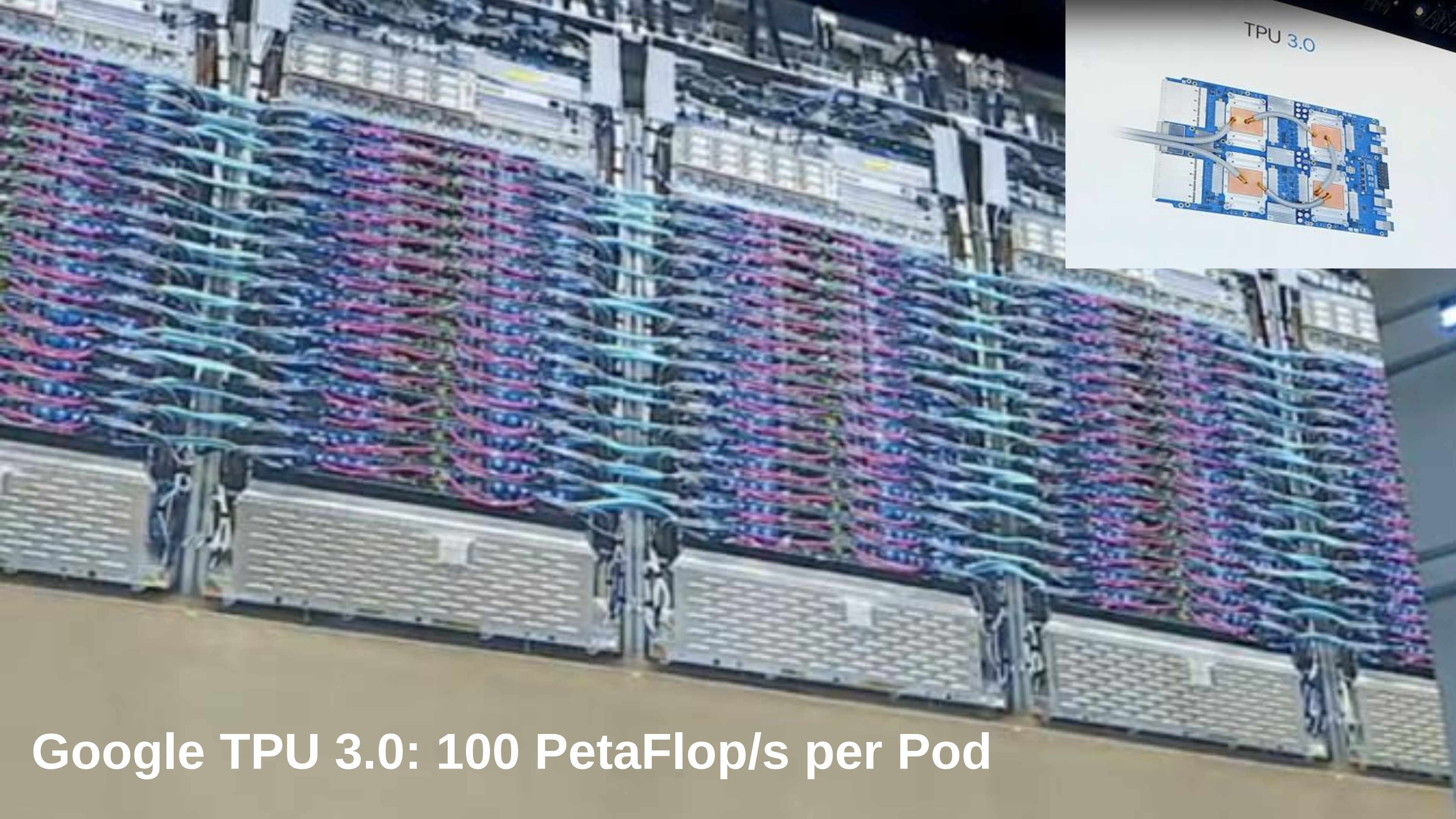
Cataloging the Visible Universe through Bayesian Inference at Petascale

Jeffrey Regier[*], Kiran Pamnany[†], Keno Fischer[‡], Andreas Noack[§], Maximilian Lam[*], Jarrett Revels[§], Steve Howard[¶], Ryan Giordano[¶], David Schlegel[‖], Jon McAuliffe[¶], Rollin Thomas[‖], Prabhat[‖]

[*]*Department of Electrical Engineering and Computer Sciences, University of California, Berkeley*
[†]*Parallel Computing Lab, Intel Corporation*
[‡]*Julia Computing*
[§]*Computer Science and AI Laboratories, Massachusetts Institute of Technology*
[¶]*Department of Statistics, University of California, Berkeley*
[‖]*Lawrence Berkeley National Laboratory*



Cori Phase II – 1.5 PetaFlop/s



Google TPU 3.0: 100 PetaFlop/s per Pod

It just works (Part I)

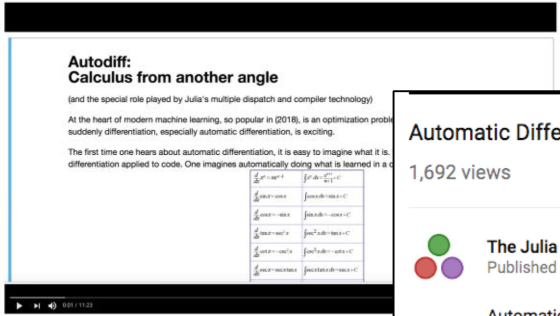
stackoverflow

importance of PCA or SVD in machine learning

31

All this time (specially in Netflix contest), I always come across this blog (or leaderboard forum) where they mention how by applying a simple SVD step on data helped them in reducing sparsity in data or in general improved the performance of their algorithm in hand. I am trying to think (since

"We can teach our autodiff system to differentiate the svd" vs "It just works because of built in abstractions in language design"



Automatic Differentiation in 10 minutes with Julia

1,692 views

The Julia Language
Published on Jun 1, 2018

Automatic differentiation is a key technique in AI - especially in deep neural networks. Here's a short video by MIT's Prof. Alan Edelman teaching automatic differentiation in 10 minutes using Julia.

SHOW MORE

julia

It just works (Part II)

Machine learning with operators (not dense matrices, not sparse matrices)

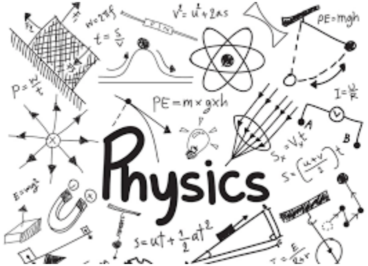
$$\begin{pmatrix} dx_2 \\ dx_3 \\ \vdots \\ dx_N \\ dx_{N+1} \end{pmatrix} = \begin{pmatrix} (x_1^T \otimes \Delta_1, \delta_1) & & & & \\ & (x_2^T \otimes \Delta_2, \delta_2) & & & \\ & & \ddots & & \\ & & & (x_{N-1}^T \otimes \Delta_{N-1}, \delta_{N-1}) & \\ & & & & (x_N^T \otimes \Delta_N, \delta_N) \end{pmatrix} \begin{pmatrix} (dW_1, db_1)^T \\ (dW_2, db_2)^T \\ \vdots \\ (dW_{N-1}, db_{N-1})^T \\ (dW_N, db_N)^T \end{pmatrix}$$

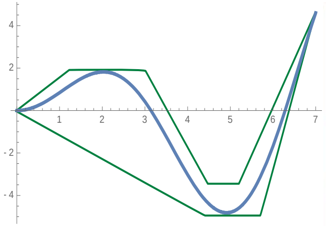
$$+ \begin{pmatrix} 0 & \ldots & 0 & 0 & 0 \\ \Delta_2 W_2 & \ldots & 0 & 0 & 0 \\ & \ddots & & & \\ & & \Delta_{N-1} W_{N-1} & 0 & 0 \\ & & & \Delta_N W_N & 0 \end{pmatrix} \begin{pmatrix} dx_2 \\ dx_3 \\ \vdots \\ dx_N \\ dx_{N+1} \end{pmatrix}$$

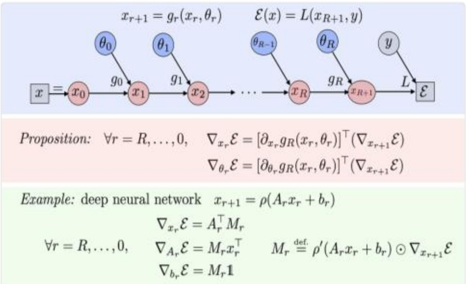
Build Operators solve with “backslash”

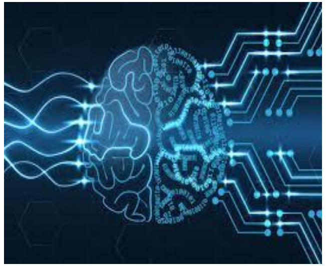
Not Blackboard → formula→ implementation→ debugging

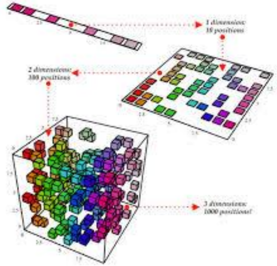
Software Toolchain











Multiphysics (PDEs)

Optimization

Adjoint Methods (Backprop/Autodiff)

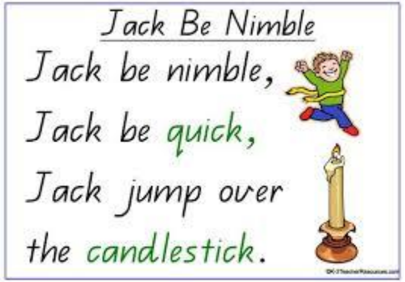
Machine Learning

Surrogat
Dim R

Composability

Sensitivity Analysis
Confidence Intervals

Performance
Nimble/Agile



?

Uncertainty
Quantification

S